Emergent Agent-Based Scheduling of Manufacturing Systems

Leslie D. Interrante, Integrated Product Development Department
Steven Y. Goldsmith, Advanced Information Systems Laboratory
Sandia National Laboratories[1]
MS 0749
Albuquerque, NM 87185-5800
phone: 505-845-8926
(ldinter@sandia.gov, sygolds@sandia.gov)

Abstract: This paper describes a spectrum of agent-based finite scheduling systems which differ along two axes of description: level of abstraction within the manufacturing system and scope of reasoning of an individual agent. Within the spectrum, the agent systems exhibit differing degrees of emergent behavior, which affects the manner in which the scheduling systems reason about the propagation of schedule changes within the manufacturing subsystem. A single, general-purpose agent architecture is described which is employed to reason about scheduling across this two-dimensional spectrum of systems. The value of this architecture is that it can be used to generate agents that exhibit a variety of levels of individual agent scoping and a variety of degrees of predictive versus reactive scheduling ability. Moreover, it enables scheduling systems with dynamic architectures that adapt their structure at runtime in response to scheduling conflicts that arise at different levels of aggregation. Examples of applications are provided for a number of the systems described by the spectrum, within the application of weapons development and production.

1.0 Introduction

The domain of manufacturing scheduling is particularly challenging from the standpoint of information processing because of the many different, interacting subsystems at a variety of levels of abstraction and because of the volatile nature of the manufacturing environment. Successful manufacturing companies can attribute much of their success to well-coordinated development and production systems. However, many of the industry successes today are based on relatively simple coordination processes (e.g., checklists; well-trained operators, managers, and technical development personnel; and inventory control techniques such as kanbans). MRP, higher-level systems such as ERP, and constraint-propagation-based scheduling systems are widely used for scheduling applications.

State-of-the-art technologies in agent-based systems and emergent behavior promise to provide more sophisticated coordination of manufacturing systems that are able to quickly respond to changing conditions and to reflect those changes at multiple levels of abstraction. Most of the work in this area is characterized by a focus on a particular level of abstraction or agent architecture applied to a particular manufacturing problem. There is to date not enough information characterizing options and tradeoffs in applying this technology in the manufacturing scheduling domain. Research in emergent systems is characterized by a shift from the paradigm of explicit, predictive modeling of interactions to a more decentralized reactive approach in which system behavior is difficult to predict without simulating the system. The propagation of schedule changes among interacting components of the manufacturing system at a variety of levels is one of the most difficult barriers to realizing success in the achievement of more sophisticated manufacturing system coordination.

This paper describes a spectrum of agent-based scheduling systems which differ along two axes of description: level of abstraction within the manufacturing system and scope of reasoning of an individual agent. Within the spectrum, the agent systems exhibit differing degrees of emergent behavior, which affects the manner in which the scheduling systems reason about the propagation

[1] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

of schedule changes within the manufacturing subsystem. A single, general-purpose agent architecture is briefly described which is employed to reason about scheduling across this two-dimensional spectrum of systems. The value of this architecture is that it can be used to generate agents that exhibit a variety of levels of individual agent scoping and a variety of degrees of predictive versus reactive scheduling ability. Moreover, it enables scheduling systems with dynamic architectures that adapt their structure at runtime in response to scheduling conflicts that arise at different levels of aggregation. Examples of agent design and behavior are provided for a number of the systems described by the spectrum, within the application of weapons development and production.

2.0 Spectrum of Agent-Based Manufacturing Scheduling Systems

Figure 1 depicts the spectrum of agent-based scheduling systems. The columns are differentiated by level of abstraction within the manufacturing system. At the machine level, agents reason about the scheduling of an individual machine on the shop floor. At the production line/cell level, a group of interacting machines is scheduled by a single agent. At the shop floor level, multiple interacting schedules among shop floor subsystems (e.g., receiving, material handling, production lines/cells, shipping, etc.) are considered. Supplier-customer relationships and aggregate schedules among plants and distribution systems are considered by an agent at the supply chain level. As the manufacturing system level of abstraction becomes higher, the scope of the scheduling problem becomes broader and the information available to agents becomes more global in nature. As one moves across this axis, the nature of the data which drives agent behavior changes.

The rows of Figure 1 are differentiated by an agent's scope of reasoning and the degree of local versus global view of the manufacturing system interactions. These factors determine the agent system's ability to perform predictive and reactive reasoning related to scheduling. In a decentralized regime, Type A agents represent system entities at different levels of aggregation: machine, production line/cell, shop floor subsystem, and plant in a supply chain. Type A agents have local scoping and reactive reasoning. A hybrid architecture that is decentralized but temporarily aggregated includes Type A agents but adds an additional agent, the Type B agent, that is created "on the fly" to perform special reasoning across a number of Type A agents when a scheduling conflict occurs. This architecture has localized predictive behavior in addition to the reactive behavior. A third architecture, also a hybrid of decentralized and aggregate regimes, includes Type A and Type B agents but incorporates a supervisory agent of Type C, which can reason about the global effect of multiple interacting schedule propagations on the overall manufacturing system goals. As the individual agent's scope of reasoning and agent system's ability to explicitly reason about scheduling interactions broadens, the message-passing burden decreases and the agent system moves from a purely reactive system to one that is more predictive in nature.

3.0 Related Work

Several researchers have examined dynamic scheduling for manufacturing and other domains. Zweben, et al. [Zweben 1992] investigated rescheduling with iterative repair for the scheduling of shuttle operations. Xiong, et al [Xiong 1992] have examined the use of intelligent backtracking for job shop scheduling. Shaw and Whinston [Shaw 1989] examined FMS scheduling with goal-directed inferencing for the sharing of machine resources. Pan, et al. [Pan 1992] have described an intelligent material handling system for computer integrated manufacturing. Wu and Wysk [Wu 1990] developed an inference structure which provides forward and goal-directed inference with simulation capabilities for control and scheduling in manufacturing. Ringer [Ringer 1992] used time phased abstractions for combining predictive and reactive scheduling. Hadavi, et al.[Hadavi 1992] have developed a recursive architecture for real-time distributed scheduling. Drummond [Drummond 1993] has investigated contingent scheduling using explicit representations of disjunctions in schedules for proactive error management. Research has been performed in the area of reactive scheduling. Smith and Morton [Smith 1993] have examined schedule revision under changing conditions with OR dispatching and AI interval scheduling. Fox has examined the

	Manufacturing System Level of Abstraction			
Agent Scope & Behavior	Machine Level	Line/Cell Level	Shop Floor Level	Supply Chain Level
Type A Physical Entities	Manages machine scheduling	Manages line/cell scheduling	Manages subsystem scheduling	Manages company scheduling
Type AB + Delta Agents	Manages scheduling for multiple machines	Manages scheduling for multiple cells	Manages scheduling for multiple subsystems	Manages scheduling for multiple companies
Type ABC + Supervisory Agents	X	Manages scheduling for a subsystem	Manages scheduling for shop floor	Manages scheduling for supply chain

Figure 1. Spectrum of Agent-Based Manufacturing Scheduling Systems

requirements for a dynamic scheduling system [Fox 1993]. Beck's system, TOSCA, manages job-shop scheduling constraints [Beck 1993]. Diaz, et al [Diaz 1991] have focused on customer order information for schedule and reschedule production in a steel plant. Burke and Prosser's DAS [Burke 1991] uses negotiation between autonomous agents to perform predictive and reactive scheduling. Recent work in emergent manufacturing systems includes [Paranuk 1993; 1994; 1996; 1997] and [Greenstein 1995].

4.0 Application Manufacturing System

The application area we consider in this paper is development and manufacturing within the realm of weapon system production. The environment is characterized by high-reliability, low-volume production in a controlled environment, with strict quality certification requirements. The application environment will be described with respect to the horizontal axis of manufacturing system level of abstraction as depicted in Figure 1.

At the machine level, one example within the domain is the brazing operation, which comprises seven of thirteen total assembly processes for the example product. Brazing is a batch operation, involving a four-hour cycle with a time-consuming and complicated setup operation, characterized by many small parts and complicated fixturing with tight tolerances.

At the production line level, additional processes include cleaning, plating, and evacuation of the product. The part routing is complex, with multiple brazing stages. The production cycle time is long (24-30 weeks), with small batch sizes.

Coordination of shop floor operations is a challenge, since three types of processing are interleaved on the same machines/processes: (1) regular production with an aggressive master production schedule); (2) development builds for a new design; and (3) experiments to support early design decisions.

At the supply chain level, materials and supply parts are nonstandard in many cases, with stringent quality requirements. Supply parts are typically special-ordered in low volume. Vendors include private industry as well as other specialized facilities. Some supplies are acquired in-house. The process of acquiring supply parts and materials is a challenge because of simultaneous regular production, development builds, and experimentation. The finished product is the primary component of a larger assembly which, upon completion, is shipped out to another facility.

5.0 General Agent Architecture

The Standard Agent Framework is an object-oriented framework that enables the exploratory development of multiagent systems that interact with human users. The Standard Agent Framework provides a means for constructing and customizing multiagent systems by specialization of base classes (architecture-driven) and by composition (data-driven). The framework comprises two associated abstract classes: *agent* and *agency*. An agency identifies an independent locus of processes, activities, and knowledge typically associated with an company, organization, department, site, household, machine, or some other natural partitioning of the application domain. The underlying assumption is that the application is naturally modeled as a group of interacting agencies. The agency provides a containing context for a collection of agents. The activities of the agency are conducted by its constituent agents. Agents inhabit an agency for the express purpose of providing *services*, including intra-agency communications, that maintain the functioning of the agency and lead to satisfaction of the ultimate objectives of the agency. An agent performs domain-specific tasks on behalf of human actors and other agents. The state of the agency object evolves in time as human actors, agents, and resources are added and deleted.

Actual agent systems are implemented by the specialization and instantiation of four concrete classes: (1) Standard Agency; (2) Standard Agent; (3) Human Actor; and (4) Resources. The class Standard Agency is an elaboration of the agency concept that includes human activities within the agency and devices for data-oriented activities such as storage and communications. An instance of Standard Agency is a persistent, identity-bearing composite object that contains collections of the component classes Standard Agent, Human Actor, and Resources. The protocols for the Standard Agency class are primarily introspective methods that return component instances to enable an overview of agency status for debug and display purposes. The class Standard Agent implements instances of agents that have specific attributes: autonomy, social ability, reflexivity, and pro-activeness [Wooldridge and Jennings 1997]. An instance of Standard Agent is a composite object. The primary Standard Agent protocols are an interface mechanism that enables interaction with other agents and human actors, a reflexive action mechanism for rapidly responding to event objects in its environment, and a generic inference mechanism for achieving explicit goals. The interaction and inference protocols can be specialized with methods that implement other agent architectures and mechanisms. Agents are self-contained threads of execution that execute both periodically and through immediate scheduling. Agent structure is recursive so agents can create and contain other agents at runtime.

Human actors are people that inhabit the agency through an interface device and interact with agents to accomplish tasks. Human actor objects are temporary objects that contain an interface address, an interface object that captures the display, data entry and control functions currently available to the person, and a persistent person object that holds personal data, passwords, email address, and an account object that provides access to past and current workspaces. A workspace object contains objects created and stored by the person during work sessions.

Agents and human actors have access to resources such as databases, fax machines, telephones, email handlers, and other useful services. Resource objects provide concurrency control and access protocols for agency resources. Subclasses of the resource class implement objects representing data bases, fax machines, printers, email ports, EDI ports and other commonplace legacy devices in the agency environment.

The Standard Agent Framework supports distributed agent systems. Agency objects may be distributed in a network environment to create a collaborative enterprise structure of interconnected agencies. The fundamental activity conducted among distributed agencies is the trading of domain objects through proxy agents that represent one agency within the agent collection of another agency. These proxy objects delegate all messages (except for a local request for identifying information about the represented agency) to the actual agent residing in the agency. Public proxies are registered in an agency network phonebook with a well-know address. To find other agencies, an agent issues one or more queries to the phonebook and is returned the proxy objects matching the query. The agent proxies interned within an agency form a persistent network of agencies. Such networks are called*durable proxy networks*.

6.0 Type A Agents

Type A agents represent system entities such as a machine, a production line/cell, the shop floor subsystem, or a plant in a supply chain. The agent system exhibits purely local scoping and completely reactive reasoning. The agent represents the following physical entity for each manufacturing system level of abstraction described in Figure 1: (1) machine level- agent represents a machine; (2) production line/cell level- agent represents a line; (3) shop floor level- agent represents a manufacturing subsystem; and (4) supply chain level - agent represents a plant, warehouse, or distribution system.

A Type A agent is aware of the state of the system entity it represents. The agent knows its local production goals and is able to compute its internal schedule to meet the goals. It knows what resources are needed to meet its schedule and can ask for commitments from other agents for resources (e.g., supply parts). The agent's "awareness" is limited to its own schedule. The agent cannot compute the effects of its schedule changes on other entities in the system; nor can it compute the effect of schedule propagations outside of its own schedule. The resolution of conflicts in such an agent system requires a high volume of message passing which follows the line of propagation of multiple schedule interactions. The scheduling system is completely reactive. Explicit predictive or learning capabilities are not present in a single agent;, and it cannot tune system performance or to try to resolve schedule contingencies in an optimized fashion. However, it is possible that through local collaborations the agent collective may achieve the same objectives as a reactive centralized scheduling system.

Type A agents implement a truly decentralized factory comprising autonomous machines (Figure 2). Scheduling and production are emergent properties of the machine collective. Each machine is responsible for adjusting its local production schedule according to its capacity and the states of its suppliers and customers. An agent architecture for an autonomous shop floor is shown in Figure 2. Each machine is represented by an agent that collaborates with its suppliers and customers. Customer agents send requests (R) for needed parts to supplier agents. Supplier agents send defeasible commitments (C) to the customer agents that identify their best effort to supply the needed parts or materials. Each request R is of the form {N, TYPE, <T1 T2>, P, X}, where N is the number of items needed, TYPE denotes the type of item (necessary when the supplier produces more than one kind of item), <T1 T2> is a closed time interval over which the items must be delivered (not earlier than T1 and not later than T2), and P is the priority ranking. The term X represents other decision criteria provided to the supplier from agents upstream that influence its scheduling process.

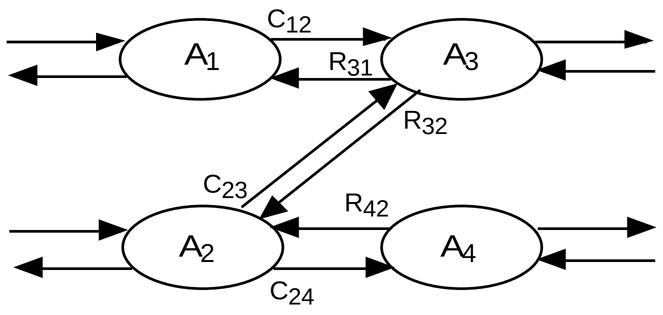

Figure 2. Autonomous Production Collective

The supplier agent computes a new schedule and replies to the request with a commitment of the form {n, Type, <t1 t2>, P, Y} where n is the promised production quantity to be delivered within interval <t1 t2> at priority P. The term Y represents decision criteria passed from supplier to customer that influences the customer's scheduling process. Commitments are honored by suppliers unless a higher priority request causes a conflict with outstanding commitments, or if a supplier of the supplier cannot honor its commitments. In Figure 2, agent (machine) A_3 is a potential competitor of agent A_4 for the products of agent A_2. When the requests for production from A_3 and A_4 are in conflict, A_2 must resolve the conflict according to a conflict resolution strategy that provides global stability. One scheme uses the priority P to select the product chain with the higher priority for service at its requested capacity. Other schemes may use weighting based on priority to allocate some services to each requester. The local conflict resolution behavior of individual agents influences the global stability and performance of the factory. When a supplier defaults on its commitment, the default may propagate forward in the supply chain causing cascading defaults. As the supply chain reorganizes, customers may also default on their requests and release suppliers from previously negotiated commitments, causing a release from commitments to be propagated backward in the supply chain. If conflicts arise that cannot be resolved through local interactions, a more global scheduling process is required. Further research is required to investigate the global properties of this autonomous supply chain, especially global stability and other macroscopic properties.

An example of a Type A agent for a single machine is the braze furnace agent. This agent knows the subassemblies it is responsible for joining and the associated parts/fixturing requirements. The number of each subassembly type which must be produced each week is known by the agent. It can compute a furnace schedule, given part/fixture commitments that it is able to obtain from other agents. If the appropriate amount of supply parts are unavailable, the agent cannot commit to producing an adequate number of brazed subassemblies for subsequent operations (i.e., meeting its given production goals). Thus, the propagation of schedule interactions occurs in terms of pairwise commitments or the lack thereof, but the system has no ability to explicitly reason about the global effect of schedule interaction propagations.

7.0 Type B Agent System

A new kind of hybrid architecture comprises Type A agents and a transient agent called a Type B agent that is created at runtime by the Type A collective to resolve a scheduling conflict. This agent system has local predictive ability in addition to the reactive behavior. A Type B agent represents the following system entities for each manufacturing system level of abstraction described in Figure 1: (1) machine level- the agent reasons about multiple machine schedules; (2) production line/cell level- the agent reasons about multiple line schedules; (3) shop floor level- the agent reasons about multiple manufacturing subsystem schedules; and (4) supply chain level- the agent reasons about